| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ThreadMXBean.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/management/ThreadInfo.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/lang/management/ThreadMXBean.html)    [**NO FRAMES**](http://docs.google.com/ThreadMXBean.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **java.lang.management**

Interface ThreadMXBean

public interface **ThreadMXBean**

The management interface for the thread system of the Java virtual machine.

A Java virtual machine has a single instance of the implementation class of this interface. This instance implementing this interface is an [MXBean](http://docs.google.com/ManagementFactory.html#MXBean) that can be obtained by calling the [ManagementFactory.getThreadMXBean()](http://docs.google.com/java/lang/management/ManagementFactory.html#getThreadMXBean()) method or from the [platform MBeanServer](http://docs.google.com/java/lang/management/ManagementFactory.html#getPlatformMBeanServer()) method.

The ObjectName for uniquely identifying the MXBean for the thread system within an MBeanServer is:

[java.lang:type=Threading](http://docs.google.com/java/lang/management/ManagementFactory.html#THREAD_MXBEAN_NAME)

#### Thread ID

Thread ID is a positive long value returned by calling the [Thread.getId()](http://docs.google.com/java/lang/Thread.html#getId()) method for a thread. The thread ID is unique during its lifetime. When a thread is terminated, this thread ID may be reused.

Some methods in this interface take a thread ID or an array of thread IDs as the input parameter and return per-thread information.

#### Thread CPU time

A Java virtual machine implementation may support measuring the CPU time for the current thread, for any thread, or for no threads.

The [isThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported()) method can be used to determine if a Java virtual machine supports measuring of the CPU time for any thread. The [isCurrentThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isCurrentThreadCpuTimeSupported()) method can be used to determine if a Java virtual machine supports measuring of the CPU time for the current thread. A Java virtual machine implementation that supports CPU time measurement for any thread will also support that for the current thread.

The CPU time provided by this interface has nanosecond precision but not necessarily nanosecond accuracy.

A Java virtual machine may disable CPU time measurement by default. The [isThreadCpuTimeEnabled()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeEnabled()) and [setThreadCpuTimeEnabled(boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadCpuTimeEnabled(boolean)) methods can be used to test if CPU time measurement is enabled and to enable/disable this support respectively. Enabling thread CPU measurement could be expensive in some Java virtual machine implementations.

#### Thread Contention Monitoring

Some Java virtual machines may support thread contention monitoring. When thread contention monitoring is enabled, the accumulated elapsed time that the thread has blocked for synchronization or waited for notification will be collected and returned in the [ThreadInfo](http://docs.google.com/ThreadInfo.html#SyncStats) object.

The [isThreadContentionMonitoringSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadContentionMonitoringSupported()) method can be used to determine if a Java virtual machine supports thread contention monitoring. The thread contention monitoring is disabled by default. The [setThreadContentionMonitoringEnabled(boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadContentionMonitoringEnabled(boolean)) method can be used to enable thread contention monitoring.

#### Synchronization Information and Deadlock Detection

Some Java virtual machines may support monitoring of [object monitor usage](http://docs.google.com/java/lang/management/ThreadMXBean.html#isObjectMonitorUsageSupported()) and [ownable synchronizer usage](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported()). The [getThreadInfo(long[], boolean, boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D,%20boolean,%20boolean)) and [dumpAllThreads(boolean, boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#dumpAllThreads(boolean,%20boolean)) methods can be used to obtain the thread stack trace and synchronization information including which [*lock*](http://docs.google.com/java/lang/management/LockInfo.html) a thread is blocked to acquire or waiting on and which locks the thread currently owns.

The ThreadMXBean interface provides the [findMonitorDeadlockedThreads()](http://docs.google.com/java/lang/management/ThreadMXBean.html#findMonitorDeadlockedThreads()) and [findDeadlockedThreads()](http://docs.google.com/java/lang/management/ThreadMXBean.html#findDeadlockedThreads()) methods to find deadlocks in the running application.

**Since:** 1.5 **See Also:** [JMX Specification.](http://docs.google.com/javax/management/package-summary.html),  [Ways to Access MXBeans](http://docs.google.com/package-summary.html#examples)

| **Method Summary** | |
| --- | --- |
| [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] | [**dumpAllThreads**](http://docs.google.com/java/lang/management/ThreadMXBean.html#dumpAllThreads(boolean,%20boolean))(boolean lockedMonitors, boolean lockedSynchronizers)            Returns the thread info for all live threads with stack trace and synchronization information. |
| long[] | [**findDeadlockedThreads**](http://docs.google.com/java/lang/management/ThreadMXBean.html#findDeadlockedThreads())()            Finds cycles of threads that are in deadlock waiting to acquire object monitors or [ownable synchronizers](http://docs.google.com/LockInfo.html#OwnableSynchronizer). |
| long[] | [**findMonitorDeadlockedThreads**](http://docs.google.com/java/lang/management/ThreadMXBean.html#findMonitorDeadlockedThreads())()            Finds cycles of threads that are in deadlock waiting to acquire object monitors. |
| long[] | [**getAllThreadIds**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getAllThreadIds())()            Returns all live thread IDs. |
| long | [**getCurrentThreadCpuTime**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getCurrentThreadCpuTime())()            Returns the total CPU time for the current thread in nanoseconds. |
| long | [**getCurrentThreadUserTime**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getCurrentThreadUserTime())()            Returns the CPU time that the current thread has executed in user mode in nanoseconds. |
| int | [**getDaemonThreadCount**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getDaemonThreadCount())()            Returns the current number of live daemon threads. |
| int | [**getPeakThreadCount**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getPeakThreadCount())()            Returns the peak live thread count since the Java virtual machine started or peak was reset. |
| int | [**getThreadCount**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadCount())()            Returns the current number of live threads including both daemon and non-daemon threads. |
| long | [**getThreadCpuTime**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadCpuTime(long))(long id)            Returns the total CPU time for a thread of the specified ID in nanoseconds. |
| [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) | [**getThreadInfo**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long))(long id)            Returns the thread info for a thread of the specified id with no stack trace. |
| [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] | [**getThreadInfo**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D))(long[] ids)            Returns the thread info for each thread whose ID is in the input array ids with no stack trace. |
| [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] | [**getThreadInfo**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D,%20boolean,%20boolean))(long[] ids, boolean lockedMonitors, boolean lockedSynchronizers)            Returns the thread info for each thread whose ID is in the input array ids, with stack trace and synchronization information. |
| [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] | [**getThreadInfo**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D,%20int))(long[] ids, int maxDepth)            Returns the thread info for each thread whose ID is in the input array ids, with stack trace of a specified number of stack trace elements. |
| [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) | [**getThreadInfo**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long,%20int))(long id, int maxDepth)            Returns a thread info for a thread of the specified id, with stack trace of a specified number of stack trace elements. |
| long | [**getThreadUserTime**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadUserTime(long))(long id)            Returns the CPU time that a thread of the specified ID has executed in user mode in nanoseconds. |
| long | [**getTotalStartedThreadCount**](http://docs.google.com/java/lang/management/ThreadMXBean.html#getTotalStartedThreadCount())()            Returns the total number of threads created and also started since the Java virtual machine started. |
| boolean | [**isCurrentThreadCpuTimeSupported**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isCurrentThreadCpuTimeSupported())()            Tests if the Java virtual machine supports CPU time measurement for the current thread. |
| boolean | [**isObjectMonitorUsageSupported**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isObjectMonitorUsageSupported())()            Tests if the Java virtual machine supports monitoring of object monitor usage. |
| boolean | [**isSynchronizerUsageSupported**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported())()            Tests if the Java virtual machine supports monitoring of  [ownable synchronizer](http://docs.google.com/LockInfo.html#OwnableSynchronizer) usage. |
| boolean | [**isThreadContentionMonitoringEnabled**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadContentionMonitoringEnabled())()            Tests if thread contention monitoring is enabled. |
| boolean | [**isThreadContentionMonitoringSupported**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadContentionMonitoringSupported())()            Tests if the Java virtual machine supports thread contention monitoring. |
| boolean | [**isThreadCpuTimeEnabled**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeEnabled())()            Tests if thread CPU time measurement is enabled. |
| boolean | [**isThreadCpuTimeSupported**](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported())()            Tests if the Java virtual machine implementation supports CPU time measurement for any thread. |
| void | [**resetPeakThreadCount**](http://docs.google.com/java/lang/management/ThreadMXBean.html#resetPeakThreadCount())()            Resets the peak thread count to the current number of live threads. |
| void | [**setThreadContentionMonitoringEnabled**](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadContentionMonitoringEnabled(boolean))(boolean enable)            Enables or disables thread contention monitoring. |
| void | [**setThreadCpuTimeEnabled**](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadCpuTimeEnabled(boolean))(boolean enable)            Enables or disables thread CPU time measurement. |

| **Method Detail** |
| --- |

### getThreadCount

int **getThreadCount**()

Returns the current number of live threads including both daemon and non-daemon threads.

**Returns:**the current number of live threads.

### getPeakThreadCount

int **getPeakThreadCount**()

Returns the peak live thread count since the Java virtual machine started or peak was reset.

**Returns:**the peak live thread count.

### getTotalStartedThreadCount

long **getTotalStartedThreadCount**()

Returns the total number of threads created and also started since the Java virtual machine started.

**Returns:**the total number of threads started.

### getDaemonThreadCount

int **getDaemonThreadCount**()

Returns the current number of live daemon threads.

**Returns:**the current number of live daemon threads.

### getAllThreadIds

long[] **getAllThreadIds**()

Returns all live thread IDs. Some threads included in the returned array may have been terminated when this method returns.

**Returns:**an array of long, each is a thread ID. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor").

### getThreadInfo

[ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) **getThreadInfo**(long id)

Returns the thread info for a thread of the specified id with no stack trace. This method is equivalent to calling:[getThreadInfo(id, 0);](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long,%20int))

This method returns a ThreadInfo object representing the thread information for the thread of the specified ID. The stack trace, locked monitors, and locked synchronizers in the returned ThreadInfo object will be empty. If a thread of the given ID is not alive or does not exist, this method will return null. A thread is alive if it has been started and has not yet died.

**MBeanServer access**:

The mapped type of ThreadInfo is CompositeData with attributes as specified in the [ThreadInfo.from](http://docs.google.com/java/lang/management/ThreadInfo.html#from(javax.management.openmbean.CompositeData)) method.

**Parameters:**id - the thread ID of the thread. Must be positive. **Returns:**a [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) object for the thread of the given ID with no stack trace, no locked monitor and no synchronizer info; null if the thread of the given ID is not alive or it does not exist. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if id <= 0. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor").

### getThreadInfo

[ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] **getThreadInfo**(long[] ids)

Returns the thread info for each thread whose ID is in the input array ids with no stack trace. This method is equivalent to calling:

[getThreadInfo](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D,%20int))(ids, 0);

This method returns an array of the ThreadInfo objects. The stack trace, locked monitors, and locked synchronizers in each ThreadInfo object will be empty. If a thread of a given ID is not alive or does not exist, the corresponding element in the returned array will contain null. A thread is alive if it has been started and has not yet died.

**MBeanServer access**:

The mapped type of ThreadInfo is CompositeData with attributes as specified in the [ThreadInfo.from](http://docs.google.com/java/lang/management/ThreadInfo.html#from(javax.management.openmbean.CompositeData)) method.

**Parameters:**ids - an array of thread IDs. **Returns:**an array of the [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) objects, each containing information about a thread whose ID is in the corresponding element of the input array of IDs with no stack trace, no locked monitor and no synchronizer info. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any element in the input array ids is <= 0. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor").

### getThreadInfo

[ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) **getThreadInfo**(long id,  
 int maxDepth)

Returns a thread info for a thread of the specified id, with stack trace of a specified number of stack trace elements. The maxDepth parameter indicates the maximum number of [StackTraceElement](http://docs.google.com/java/lang/StackTraceElement.html) to be retrieved from the stack trace. If maxDepth == Integer.MAX\_VALUE, the entire stack trace of the thread will be dumped. If maxDepth == 0, no stack trace of the thread will be dumped. This method does not obtain the locked monitors and locked synchronizers of the thread.

When the Java virtual machine has no stack trace information about a thread or maxDepth == 0, the stack trace in the ThreadInfo object will be an empty array of StackTraceElement.

If a thread of the given ID is not alive or does not exist, this method will return null. A thread is alive if it has been started and has not yet died.

**MBeanServer access**:

The mapped type of ThreadInfo is CompositeData with attributes as specified in the [ThreadInfo.from](http://docs.google.com/java/lang/management/ThreadInfo.html#from(javax.management.openmbean.CompositeData)) method.

**Parameters:**id - the thread ID of the thread. Must be positive.maxDepth - the maximum number of entries in the stack trace to be dumped. Integer.MAX\_VALUE could be used to request the entire stack to be dumped. **Returns:**a [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) of the thread of the given ID with no locked monitor and synchronizer info. null if the thread of the given ID is not alive or it does not exist. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if id <= 0. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if maxDepth is negative. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor").

### getThreadInfo

[ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] **getThreadInfo**(long[] ids,  
 int maxDepth)

Returns the thread info for each thread whose ID is in the input array ids, with stack trace of a specified number of stack trace elements. The maxDepth parameter indicates the maximum number of [StackTraceElement](http://docs.google.com/java/lang/StackTraceElement.html) to be retrieved from the stack trace. If maxDepth == Integer.MAX\_VALUE, the entire stack trace of the thread will be dumped. If maxDepth == 0, no stack trace of the thread will be dumped. This method does not obtain the locked monitors and locked synchronizers of the threads.

When the Java virtual machine has no stack trace information about a thread or maxDepth == 0, the stack trace in the ThreadInfo object will be an empty array of StackTraceElement.

This method returns an array of the ThreadInfo objects, each is the thread information about the thread with the same index as in the ids array. If a thread of the given ID is not alive or does not exist, null will be set in the corresponding element in the returned array. A thread is alive if it has been started and has not yet died.

**MBeanServer access**:

The mapped type of ThreadInfo is CompositeData with attributes as specified in the [ThreadInfo.from](http://docs.google.com/java/lang/management/ThreadInfo.html#from(javax.management.openmbean.CompositeData)) method.

**Parameters:**ids - an array of thread IDsmaxDepth - the maximum number of entries in the stack trace to be dumped. Integer.MAX\_VALUE could be used to request the entire stack to be dumped. **Returns:**an array of the [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) objects, each containing information about a thread whose ID is in the corresponding element of the input array of IDs with no locked monitor and synchronizer info. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if maxDepth is negative. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if any element in the input array ids is <= 0. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor").

### isThreadContentionMonitoringSupported

boolean **isThreadContentionMonitoringSupported**()

Tests if the Java virtual machine supports thread contention monitoring.

**Returns:**true if the Java virtual machine supports thread contention monitoring; false otherwise.

### isThreadContentionMonitoringEnabled

boolean **isThreadContentionMonitoringEnabled**()

Tests if thread contention monitoring is enabled.

**Returns:**true if thread contention monitoring is enabled; false otherwise. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support thread contention monitoring.**See Also:**[isThreadContentionMonitoringSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadContentionMonitoringSupported())

### setThreadContentionMonitoringEnabled

void **setThreadContentionMonitoringEnabled**(boolean enable)

Enables or disables thread contention monitoring. Thread contention monitoring is disabled by default.

**Parameters:**enable - true to enable; false to disable. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support thread contention monitoring. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("control").**See Also:**[isThreadContentionMonitoringSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadContentionMonitoringSupported())

### getCurrentThreadCpuTime

long **getCurrentThreadCpuTime**()

Returns the total CPU time for the current thread in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy. If the implementation distinguishes between user mode time and system mode time, the returned CPU time is the amount of time that the current thread has executed in user mode or system mode.

This is a convenient method for local management use and is equivalent to calling:

[getThreadCpuTime](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadCpuTime(long))(Thread.currentThread().getId());

**Returns:**the total CPU time for the current thread if CPU time measurement is enabled; -1 otherwise. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support CPU time measurement for the current thread.**See Also:**[getCurrentThreadUserTime()](http://docs.google.com/java/lang/management/ThreadMXBean.html#getCurrentThreadUserTime()), [isCurrentThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isCurrentThreadCpuTimeSupported()), [isThreadCpuTimeEnabled()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeEnabled()), [setThreadCpuTimeEnabled(boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadCpuTimeEnabled(boolean))

### getCurrentThreadUserTime

long **getCurrentThreadUserTime**()

Returns the CPU time that the current thread has executed in user mode in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy.

This is a convenient method for local management use and is equivalent to calling:

[getThreadUserTime](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadUserTime(long))(Thread.currentThread().getId());

**Returns:**the user-level CPU time for the current thread if CPU time measurement is enabled; -1 otherwise. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support CPU time measurement for the current thread.**See Also:**[getCurrentThreadCpuTime()](http://docs.google.com/java/lang/management/ThreadMXBean.html#getCurrentThreadCpuTime()), [isCurrentThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isCurrentThreadCpuTimeSupported()), [isThreadCpuTimeEnabled()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeEnabled()), [setThreadCpuTimeEnabled(boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadCpuTimeEnabled(boolean))

### getThreadCpuTime

long **getThreadCpuTime**(long id)

Returns the total CPU time for a thread of the specified ID in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy. If the implementation distinguishes between user mode time and system mode time, the returned CPU time is the amount of time that the thread has executed in user mode or system mode.

If the thread of the specified ID is not alive or does not exist, this method returns -1. If CPU time measurement is disabled, this method returns -1. A thread is alive if it has been started and has not yet died.

If CPU time measurement is enabled after the thread has started, the Java virtual machine implementation may choose any time up to and including the time that the capability is enabled as the point where CPU time measurement starts.

**Parameters:**id - the thread ID of a thread **Returns:**the total CPU time for a thread of the specified ID if the thread of the specified ID exists, the thread is alive, and CPU time measurement is enabled; -1 otherwise. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if id <= 0 . [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support CPU time measurement for other threads.**See Also:**[getThreadUserTime(long)](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadUserTime(long)), [isThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported()), [isThreadCpuTimeEnabled()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeEnabled()), [setThreadCpuTimeEnabled(boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadCpuTimeEnabled(boolean))

### getThreadUserTime

long **getThreadUserTime**(long id)

Returns the CPU time that a thread of the specified ID has executed in user mode in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy.

If the thread of the specified ID is not alive or does not exist, this method returns -1. If CPU time measurement is disabled, this method returns -1. A thread is alive if it has been started and has not yet died.

If CPU time measurement is enabled after the thread has started, the Java virtual machine implementation may choose any time up to and including the time that the capability is enabled as the point where CPU time measurement starts.

**Parameters:**id - the thread ID of a thread **Returns:**the user-level CPU time for a thread of the specified ID if the thread of the specified ID exists, the thread is alive, and CPU time measurement is enabled; -1 otherwise. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if id <= 0 . [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support CPU time measurement for other threads.**See Also:**[getThreadCpuTime(long)](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadCpuTime(long)), [isThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported()), [isThreadCpuTimeEnabled()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeEnabled()), [setThreadCpuTimeEnabled(boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#setThreadCpuTimeEnabled(boolean))

### isThreadCpuTimeSupported

boolean **isThreadCpuTimeSupported**()

Tests if the Java virtual machine implementation supports CPU time measurement for any thread. A Java virtual machine implementation that supports CPU time measurement for any thread will also support CPU time measurement for the current thread.

**Returns:**true if the Java virtual machine supports CPU time measurement for any thread; false otherwise.

### isCurrentThreadCpuTimeSupported

boolean **isCurrentThreadCpuTimeSupported**()

Tests if the Java virtual machine supports CPU time measurement for the current thread. This method returns true if [isThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported()) returns true.

**Returns:**true if the Java virtual machine supports CPU time measurement for current thread; false otherwise.

### isThreadCpuTimeEnabled

boolean **isThreadCpuTimeEnabled**()

Tests if thread CPU time measurement is enabled.

**Returns:**true if thread CPU time measurement is enabled; false otherwise. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support CPU time measurement for other threads nor for the current thread.**See Also:**[isThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported()), [isCurrentThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isCurrentThreadCpuTimeSupported())

### setThreadCpuTimeEnabled

void **setThreadCpuTimeEnabled**(boolean enable)

Enables or disables thread CPU time measurement. The default is platform dependent.

**Parameters:**enable - true to enable; false to disable. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support CPU time measurement for any threads nor for the current thread. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("control").**See Also:**[isThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isThreadCpuTimeSupported()), [isCurrentThreadCpuTimeSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isCurrentThreadCpuTimeSupported())

### findMonitorDeadlockedThreads

long[] **findMonitorDeadlockedThreads**()

Finds cycles of threads that are in deadlock waiting to acquire object monitors. That is, threads that are blocked waiting to enter a synchronization block or waiting to reenter a synchronization block after an [Object.wait](http://docs.google.com/java/lang/Object.html#wait(long)) call, where each thread owns one monitor while trying to obtain another monitor already held by another thread in a cycle.

More formally, a thread is *monitor deadlocked* if it is part of a cycle in the relation "is waiting for an object monitor owned by". In the simplest case, thread A is blocked waiting for a monitor owned by thread B, and thread B is blocked waiting for a monitor owned by thread A.

This method is designed for troubleshooting use, but not for synchronization control. It might be an expensive operation.

This method finds deadlocks involving only object monitors. To find deadlocks involving both object monitors and [ownable synchronizers](http://docs.google.com/LockInfo.html#OwnableSynchronizer), the [findDeadlockedThreads](http://docs.google.com/java/lang/management/ThreadMXBean.html#findDeadlockedThreads()) method should be used.

**Returns:**an array of IDs of the threads that are monitor deadlocked, if any; null otherwise. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor").**See Also:**[findDeadlockedThreads()](http://docs.google.com/java/lang/management/ThreadMXBean.html#findDeadlockedThreads())

### resetPeakThreadCount

void **resetPeakThreadCount**()

Resets the peak thread count to the current number of live threads.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("control").**See Also:**[getPeakThreadCount()](http://docs.google.com/java/lang/management/ThreadMXBean.html#getPeakThreadCount()), [getThreadCount()](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadCount())

### findDeadlockedThreads

long[] **findDeadlockedThreads**()

Finds cycles of threads that are in deadlock waiting to acquire object monitors or [ownable synchronizers](http://docs.google.com/LockInfo.html#OwnableSynchronizer). Threads are *deadlocked* in a cycle waiting for a lock of these two types if each thread owns one lock while trying to acquire another lock already held by another thread in the cycle.

This method is designed for troubleshooting use, but not for synchronization control. It might be an expensive operation.

**Returns:**an array of IDs of the threads that are deadlocked waiting for object monitors or ownable synchronizers, if any; null otherwise. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor"). [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the Java virtual machine does not support monitoriing of ownable synchronizer usage.**Since:** 1.6 **See Also:**[isSynchronizerUsageSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported()), [findMonitorDeadlockedThreads()](http://docs.google.com/java/lang/management/ThreadMXBean.html#findMonitorDeadlockedThreads())

### isObjectMonitorUsageSupported

boolean **isObjectMonitorUsageSupported**()

Tests if the Java virtual machine supports monitoring of object monitor usage.

**Returns:**true if the Java virtual machine supports monitoring of object monitor usage; false otherwise.**Since:** 1.6 **See Also:**[dumpAllThreads(boolean, boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#dumpAllThreads(boolean,%20boolean))

### isSynchronizerUsageSupported

boolean **isSynchronizerUsageSupported**()

Tests if the Java virtual machine supports monitoring of  [ownable synchronizer](http://docs.google.com/LockInfo.html#OwnableSynchronizer) usage.

**Returns:**true if the Java virtual machine supports monitoring of ownable synchronizer usage; false otherwise.**Since:** 1.6 **See Also:**[dumpAllThreads(boolean, boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#dumpAllThreads(boolean,%20boolean))

### getThreadInfo

[ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] **getThreadInfo**(long[] ids,  
 boolean lockedMonitors,  
 boolean lockedSynchronizers)

Returns the thread info for each thread whose ID is in the input array ids, with stack trace and synchronization information.

This method obtains a snapshot of the thread information for each thread including:

* the entire stack trace,
* the object monitors currently locked by the thread if lockedMonitors is true, and
* the  [ownable synchronizers](http://docs.google.com/LockInfo.html#OwnableSynchronizer) currently locked by the thread if lockedSynchronizers is true.

This method returns an array of the ThreadInfo objects, each is the thread information about the thread with the same index as in the ids array. If a thread of the given ID is not alive or does not exist, null will be set in the corresponding element in the returned array. A thread is alive if it has been started and has not yet died.

If a thread does not lock any object monitor or lockedMonitors is false, the returned ThreadInfo object will have an empty MonitorInfo array. Similarly, if a thread does not lock any synchronizer or lockedSynchronizers is false, the returned ThreadInfo object will have an empty LockInfo array.

When both lockedMonitors and lockedSynchronizers parameters are false, it is equivalent to calling:

[getThreadInfo(ids, Integer.MAX\_VALUE)](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D,%20int))

This method is designed for troubleshooting use, but not for synchronization control. It might be an expensive operation.

**MBeanServer access**:

The mapped type of ThreadInfo is CompositeData with attributes as specified in the [ThreadInfo.from](http://docs.google.com/java/lang/management/ThreadInfo.html#from(javax.management.openmbean.CompositeData)) method.

**Parameters:**ids - an array of thread IDs.lockedMonitors - if true, retrieves all locked monitors.lockedSynchronizers - if true, retrieves all locked ownable synchronizers. **Returns:**an array of the [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) objects, each containing information about a thread whose ID is in the corresponding element of the input array of IDs. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor"). [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) -

* if lockedMonitors is true but the Java virtual machine does not support monitoring of [object monitor usage](http://docs.google.com/java/lang/management/ThreadMXBean.html#isObjectMonitorUsageSupported()); or
* if lockedSynchronizers is true but the Java virtual machine does not support monitoring of [ownable synchronizer usage](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported()).

**Since:** 1.6 **See Also:**[isObjectMonitorUsageSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isObjectMonitorUsageSupported()), [isSynchronizerUsageSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported())

### dumpAllThreads

[ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html)[] **dumpAllThreads**(boolean lockedMonitors,  
 boolean lockedSynchronizers)

Returns the thread info for all live threads with stack trace and synchronization information. Some threads included in the returned array may have been terminated when this method returns.

This method returns an array of [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) objects as specified in the [getThreadInfo(long[], boolean, boolean)](http://docs.google.com/java/lang/management/ThreadMXBean.html#getThreadInfo(long%5B%5D,%20boolean,%20boolean)) method.

**Parameters:**lockedMonitors - if true, dump all locked monitors.lockedSynchronizers - if true, dump all locked ownable synchronizers. **Returns:**an array of [ThreadInfo](http://docs.google.com/java/lang/management/ThreadInfo.html) for all live threads. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and the caller does not have ManagementPermission("monitor"). [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) -

* if lockedMonitors is true but the Java virtual machine does not support monitoring of [object monitor usage](http://docs.google.com/java/lang/management/ThreadMXBean.html#isObjectMonitorUsageSupported()); or
* if lockedSynchronizers is true but the Java virtual machine does not support monitoring of [ownable synchronizer usage](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported()).

**Since:** 1.6 **See Also:**[isObjectMonitorUsageSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isObjectMonitorUsageSupported()), [isSynchronizerUsageSupported()](http://docs.google.com/java/lang/management/ThreadMXBean.html#isSynchronizerUsageSupported())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ThreadMXBean.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/management/ThreadInfo.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/lang/management/ThreadMXBean.html)    [**NO FRAMES**](http://docs.google.com/ThreadMXBean.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).